Sprache – Kommunikation – Informatik

Akten des 26. Linguistischen Kolloquiums,
Poznań 1991

Band 2

Herausgegeben von
Józef Darski und Zygmunt Vetulani

Sonderdruck
aus LA 294

Max Niemeyer Verlag
Tübingen 1993

Andrea Tettamanzi

Equational grammars

1. Grammars à la Chomsky, rewriting and equations

In the theory of formal languages the well known notion of a grammar comes with a four–tuple (N, T, P, S), where N is a set of variable or "non–terminal" symbols, T is an alphabet or a set of "terminal" symbols, S is a non–terminal symbol which stands for "sentence" and finally P is a set of productions, that means a set of rules that can be used to rewrite strings consisting of terminal and/or non terminal symbols.

Thus, strictly speaking, a grammar is a rewriting system; the objects that are rewritten fall into two sorts N and T. One can think of arbitrary sequences of symbols from N and T as terms constructed with an "invisible" concatenation two–place associative operator.

Because there is a well–studied and well understood class of rewriting systems, namely term rewriting systems, it would be desirable, although not really necessary, to be able to view a grammar as a term rewriting system.

This is easely achieved with the introduction of a "lexicon". For the moment think of a lexicon as a one–to–one function mapping each and every element in T into a new non–terminal symbol, so that, given any string of terminal symbols, we can construct a corresponding string consisting only of non–terminal symbols. Then replace throughout the rules in P each occurrence of any terminal symbol with the relevant new non–terminal symbol according to the lexicon.

After this transformation P is a rewriting system over terms having non–terminal symbols for constants and the concatenation operator as their sole functor.

Now consider what happens if we replace the arrow in every rule of such a term rewriting system: we obtain a finite set of equations which defines an equational theory over the terms having non–terminal symbols as constants and concatenation as functor. With respect to such a theory all terms which construe a sentence belonging to the language described by the associated grammar form a unique equivalence class. This is because any term construing a sentence must eventually be equal to S in that theory.

It becomes evident at this point the analogy between derivation of a sentence in a grammar and simplification of a term in an equational theory. This concept is central to this paper: we will use it to suggest how to use techniques from equational logic theorem proving in order to deal with the parsing and generation problems.

2. Adding an internal structure to non–terminal symbols

As far as grammars à la Chomsky are concerned, the parallel between grammar and equational logic is not very exciting. It seems that there is little use in a theory which is only able to prove that any legal sentence is equal to any other legal one. In fact this is

due to the very nature of grammars like those, that have little concern on the internal structure of a sentence: what matters is just if a sentence is in the language or not.

The need for a richer description of linguistic items was felt in two areas involved with different kinds of languages. On one hand compiler designers turned their attention to attribute grammars (Knuth 1968) and, on the other, computational linguists invented unification grammars (Schieber 1986), which make use of objects called feature structures by some authors and with several other names by others.

Those formalisms share the common idea that non–terminal symbols, instead of being monadic entities, may feature an internal structure based on attributes and their values.

We have defined the concept of abstract attribute structure as a generalization of such objects.

2.1. Notation

Let $f \in Pfun(A, B)$ be a partial function: we will denote an element $a \in A$ being mapped into $b \in B$ by joining a and b with a horizontal slash:

$$a-b$$

Let $a_i \in A, i = 1, ..., n$ all and the only elements over which f is defined and $b_i \in B, i = 1, ..., n$ such elements that $f(a_i) = b_i$. The following will describe graphically the function f:

$$\left|\begin{array}{l} a_1-b_1 \\ a_2-b_2 \\ \ldots \\ a_n-b_n \end{array}\right|$$

It is understood that f is undefined over elements in A other than those apppearing in its description.

With a slight (but substantial) abuse of notation we shall use the same symbol $\perp$ to indicate both the undefined value and the function undefined everywhere.

2.2. Definition of abstract attribute structure

Let $\mathbf{A}$ and $\mathbf{C}$ be two sets. We speak of "abstract" attribute structures inasmuch as these two sets are left undetermined; only when we specify their nature we are dealing with some particular "concrete" attribute structure. Nevertheless the algebra of attribute structures does not depend on the choice of $\mathbf{A}$ and $\mathbf{C}$.

Let us suppose, without loss of generality, that T is not an element of $\mathbf{C}$; let $\{\mathbf{S}_i\}$ be a succession of sets defined as follows:

$$\mathbf{S}_0 = Pfun(\mathbf{A}, \mathbf{C} \cup \{T\})$$
$$\mathbf{S}_1 = Pfun(\mathbf{A}, \mathbf{S}_0 \cup \mathbf{C} \cup \{T\})$$
$$\mathbf{A}_2 = Pfun(\mathbf{A}, \mathbf{S}_1 \cup S_0 \cup \mathbf{C} \cup \{T\})$$
$$\ldots$$
$$\mathbf{S}_i = Pfun(\mathbf{A}, \mathbf{S}_i \cup \mathbf{S}_{i-1} \cup \ldots \cup \mathbf{S}_1 \cup \mathbf{S}_0 \cup \mathbf{C} \cup \{T\})$$
$$\ldots$$

Let **S** be the limit of this succession and let $\mathbf{V} = \mathbf{S} \cup \mathbf{C} \cup \{\mathsf{T}\}$. The elements in **A** will be called **attributes**, the elements in **V** **attribute structures**, which are divided into **atomic**, that is, contained in **C**, and **complex**, that is, contained in $\mathbf{S} \cup \{\mathsf{T}\}$.

The attribute structure $\perp \in \mathbf{S}$ is undefined over the whole set **A**. This means that for each a **A**, $\perp(a) = \perp$. The attribute structure T may be thought as the equivalence class containing all the "inconsistent" attribute structures. For T too it holds that for any $a \in \mathbf{A}$, $\mathsf{T}(a) = \mathsf{T}$. In addition we assert the following

Postulate: $\supset x \in \mathbf{V}, \supseteq a \in \mathbf{A}, x(a) = \mathsf{T} \Rightarrow x = \mathsf{T}$

Meaning that if an attribute structure maps even one attribute into the inconsistent attribute structure, then it is to be considered inconsistent on its whole.

2.3. The structure of **V**

Definition: given $x, y \in V$, we say that x is **less defined** than y, written $x \leq y$, iff:

- $x = \perp$ or $y = \mathsf{T}$ or both;
- In the general case:

$$\supset a \in A, \quad x(a) \leq y(a) \qquad \text{with } x, y \in \mathbf{S}$$

$$x = y \qquad \text{with } x, y \in \mathbf{C}$$

If neither $x \leq y$ nor $y \leq x$, we say that x and y are not comparable.

It is easy to verify that the "$\leq$" (lesser definition) relation enjoys the reflexive, transitive and antisymmetric properties, and thus is an order relation.

The "$\leq$" relation induces a partial ordering over **V**. Note that, by the definition of "$\leq$", $\perp$ is the bottom element in **V** and T is the top element. Thus **V** is a complete lattice with respect to lesser definition, with the usual two operations "$\wedge$" (meet) and "$\vee$" (join) obeying the usual lattice laws of idempotency, commutativity, associativity and absorption (see any text on Algebra).

In addition $x \leq y$ is equivalent to each of the conditions

$$x \wedge y = x \quad e \qquad x \vee y = y \qquad \text{(Consistency)}$$

and therefore

5) $x \wedge \perp = \perp \qquad x \vee \perp = x$
6) $x \wedge \mathsf{T} = x \qquad x \vee \mathsf{T} = \mathsf{T}.$

The following theorems allow to actually calculate the meet and join of any two given attribute structures.

Theorem I. Given f, g ∈S, a ∈A,
1) $[f \wedge g](a) = f(a) \wedge g(a)$
2) $[f \vee g](a) = f(a) \vee g(a)$

Proof. 1) By definition of meet, $f \wedge g \leq f$ and $f \wedge g \leq g$. This implies, by definition of "≤", that $[f \wedge g](a) \leq f(a)$ and $[f \wedge g](a) \leq g(a)$; thus $[f \wedge g](a)$ is a lower bound for f(a) and g(a). Now, for $[f \wedge g](a)$ to be the meet of f(a) and g(a), any lower bound b for f(a) and g(a) must satisfy the condition $b \leq [f \wedge g](a)$. Suppose on the contrary that there exists a b such that $b > [f \wedge g](a)$; but then there would exist a lower bound h for f and g, defined thus:

$h(a) = b$
$h(a') = [f \wedge g](a')$ for any attribute $a' \neq a$,

such that $h > f \wedge g$, contradicting the fact that $f \wedge g$ is the most definite of all the lower bounds for f and g. Therefore $[f \wedge g](a)$ is the meet of f(a) and g(a).□
The proof of 2) is analogous.

Theorem II. For any $t \in \mathbf{C}$, T covers t and t covers ⊥.

Proof. Since **T** is the top element of **V**, $t < T$. Also $\perp < t$ because ⊥ is the bottom element of **V**. Besides, by definition $t \leq s$ iff $t = s$, so there can be no attribute structure $s \neq t$ between ⊥ and t or between t and T.□

As a consequence of the above proposition, we have that
1) $t \wedge t' = \perp$ for $t \neq t'$
2) $t \vee t' = T$ for $t \neq t'$
and it is easely verified that, for any $t \in C$, $x \in S$, $x \neq \perp$, $x \neq T$,
3) $t \wedge x = \perp$
4) $t \vee x = T$

It can be shown that **V** is a modular non–distributive lattice. Since the proof is long and little interesting, it will be omitted.

2.4. Concrete attribute structures

As it was said before, we can speak of concrete attribute structures only when the set **A** of attributes and the set **C** of atomic attribute structures are defined.

A suitable definition for **A** in most practical cases consists of choosing an alphabet Σ and setting $\mathbf{A} = \Sigma^+$. If $\Sigma = \{a, b, c, ..., z\}$, **A** will contain meaningful sequences of letters. Another interesting assignment to **A** is the set **N** of natural numbers, suggesting an interpretation of attribute structures as infinitary functors. According to this interpretation, a complex attribute structure is a term constructed by the functor symbol $\mathbf{S_A}$ with an infinite arity; every attribute i is the place marker of one argument of $\mathbf{S_A}$: the attribute structure mapped into by i is the i–eth argument of $\mathbf{S_A}$.

Given a set **S**, define the bijection @: $\mathbf{S} \longrightarrow \mathbf{C}$, which maps every element $s \in \mathbf{S}$ into one atomic attribute structure @(s). If **S** features an algebraic structure, for each oper-

ator f_S of arity n defined over elements in S define an operator f_C of same arity over **C** such that, with $s_1, ..., s_n \in \mathbf{S}$, $f_C[@(s_1), ..., @(s_n)] = @[f_S(s_1, ..., s_n)]$, thus establishing an isomorphism between **S** and **C**.

Of course, nothing forbids to define **C** on the basis of several sets, each having possibly an algebraic structure, provided that the operators be properly defined.

3. Equational grammars

Attribute structures are, in a broad sense, representations of linguistic knowledge. In fact, what we expect a grammar should do is to model linguistic knowledge, much alike a theory in Physics models knowledge about physical world. So it seems quite reasonable to say that a theory defined by a set of equational axioms over the universal algebra $\mathbf{G_E} = <\mathbf{V},F>$ (see e.g. Burris, Sankappanavar 1981), with at least $(\wedge, 2), (\vee, 2) \in F$, is a grammar. Such an equational theory satisfies by construction the hypotheses of the Birkhoff theorem (Burris, Sankappanavar 1981) and thus is complete.

Provided we are able to associate at least an expression of $\mathbf{G_E}$ to any sentence of a given language, the parsing (or generation) of a sentence according to a theory E is a (finite) chain of equalities $t_1 = ... = t_n$, such that, for each step, $E \mid = t_i = t_{i+1}$.

In order to do that, define a lexicon as follows:

Definition: A **lexicon** for a language L is a binary relation $\Lambda \subseteq S \times V$, where Σ is the alphabet of L.

We can now find, for each word in a sentence, an associated attribute structure; the attribute structured thus obtained shall be connected with appropriate operators in F yielding an expression of $\mathbf{G_E}$ as we wanted.

In general there can be more than one attribute structure associated with a given word, but sometimes it happens that for any $w \in \Sigma$ there exists one and only one $x \in \mathbf{V}$ such that $\Lambda(w, x)$. In this case we are authorized to use a function "": $\Sigma \longrightarrow \mathbf{V}$, such that "w" = x whenever $\Lambda(w, x)$.

Once an expression s over $\mathbf{G_E}$ is associated with an object language sentence, the parsing of such a sentence is achieved by determining the normal forms of s with respect to the rewrite system obtained by orientation of the equations that make up the grammar; these normal form represent the possible parses of the sentence with respect to the given grammar.

4. Acknowledgments

I am personally indebted to Prof. Gianni Degli Antoni, who has promoted and driven this work with a continuous interaction and a flow of hints and advices. I would like to thank Dr. Roberto Grande for his assistance and Dr. Cinzia Baiocco for her work. I am also grateful to Prof. Nicoletta Sabadini for her kind help and criticism, as well as to Prof. Alberto Bertoni and Prof. Giancarlo Mauri for their precious concern.

References

Burris, S. / Sankappanavar, H. P. (1981): A Course in Universal Algebra. – Springer.
Hopcroft, J. E. / Ullman J. D., (1969): Formal Languages and their Relations with Automata.
Knuth, D. (1968): Semantics of CFL's.
Schieber, S. M. (1986): An Introduction to Unification–Based Approaches to Gramar. – Leland Stanford Jr Univ.